CENTRE DE RECHERCHE
SUR LES TRANSPORTS
(C.R.T.)
CENTRE FOR RESEARCH
ON TRANSPORTATION

UNIVERSITÉ DE MONTRÉAL
ÉCOLE DES HAUTES ÉTUDES COMMERCIALES
ÉCOLE POLYTECHNIQUE

CENTRE DE RECHERCHE SUR LES TRANSPORTS
UNIVERSITÉ DE MONTRÉAL
C.P. 6128, SUCCURSALE A,
MONTRÉAL
CANADA H3C 3J7

TÉLÉPHONE: (514) 343-7575
TÉLEX: (21) 05-561325
EMAIL: crt@crt.umontreal.ca
TÉLÉCOPIEUR (FAX): (514) 343-7121

CRT-802

A Truncated-Newton Method for Solving the Spatial Price Equilibrium Problem

by
Michael Florian[1,2]
S.J. Thomas[1]
and
R.V.M. Zahar[1]

[1] Département d'informatique et de recherche opérationnelle, Université de Montréal, C.P. 6128, Succursale A, Montréal, Canada H3C 3J7.

[2] Centre de recherche sur les transports, Université de Montréal, C.P. 6128, Succursale A, Montréal, Canada H3C 3J7.

Centre de recherche sur les transports — Publication #802
Département d'informatique et de recherche opérationnelle — Publication #795

November 1991

92-02304/

Dépôt légal – 4ème trimestre 1991

ABSTRACT

A truncated-Newton method is applied to the spatial price equilibrium problem and the method is compared with other specialized techniques which exploit the inherent network structure of the problem. Variants of the linear conjugate gradient algorithm for approximately solving the Newton equations are discussed. It is shown that by a judicious choice of linear iterative solver, the potential numerical difficulties associated with an indefinite or ill-conditioned Hessian matrix can be avoided. Numerical results for the Newton method are compared with those of a block-SOR method. Tests indicate that the two approaches are comparable when transportation costs are linear. A truncated-Newton algorithm is far superior if the transportation costs are nonlinear.

Keywords: spatial price equilibrium, Newton method, conjugate gradient.

- - - - - - - - - - - -

RÉSUMÉ

Une méthode Newton-tronquée est appliquée au problème d'équilibre spatial et cette méthode est comparée avec d'autres techniques spécialisées qui prennent davantage de la structure réseau du problème. Plusieurs variantes de l'algorithme des gradients conjugés pour la résolution approximative des équations de Newton sont présentées. Nous démontrons que les problèmes numériques, associés avec une matrice hessienne indéfinie ou mal conditionnée peuvent être évités. Nos résultats numériques pour la méthode de Newton sont comparés avec une méthode Newton-SOR en blocs. Ces expériences numériques indiquent que les deux approches sont comparables lorsque les coûts de transport sont linéaires. Une méthode Newton-tronquée est plus efficace si les coûts de transport sont non linéaires.

Mots-clés: équilibre spatial, méthode de Newton, gradient conjugué.

1. The Spatial Price Equilibrium Problem. In this article we describe a second derivative based optimization method for the solution of a class of spatial price equilibrium problems (SPEP) which arise in the study of market equilibria in operations research. The general market equilibrium model is originally attributed to the work of Samuelson [17] and Takayama and Judge [19], [20]. We consider the reformulation of the single commodity model proposed by Florian and Los [2] in which a convex function of supply, demand, and transportation costs, defined over a bipartite network, is minimized. The variables, representing origin to destination flows, are subject only to nonnegativity constraints. We consider a class of problems which are 'separable' and may contain nonlinear cost functions.

A description of the nonlinear SPEP model is as follows. We are given a bipartite graph consisting of m supply (output) nodes and n demand (input) nodes, resulting in mn regional transshipments. The quantity Y_i represents the total supply at the origin (market) i, where $i \in I = \{ 1, \ldots, m \}$ and Z_j represents the total demand at the destination (market) $j \in J = \{ 1, \ldots, n \}$. The cost of producing Y_i at node i is given by the supply function $S_i(Y_i)$ whereas the cost of consuming Z_j at node j is given by the inverse demand function $I_j(Z_j)$. It is assumed that $S_i(Y_i)$ is a continuous bounded monotone increasing function of $Y_i \geq 0$ and that $I_j(Z_j)$ is a continuous bounded monotone decreasing function of $Z_j \geq 0$. The amount of product x_{ij} shipped between two adjacent nodes i and j has transportation cost $c_{ij}(x_{ij})$, where c_{ij} is a continuous, monotone nondecreasing function for each pair $(i, j) \in I \times J$. This function may be concave to reflect economies of scale.

A state of equilibrium exists in the network when the cost of consumption equals the sum of production and transportation costs for all nodes in $I \times J$ where trade occurs. If trade does not occur between a pair of nodes, then the cost of consumption does not exceed the sum of production and transportation costs. These conditions may be stated mathematically as:

$$\begin{aligned} S_i(Y_i) + c_{ij}(x_{ij}) - I_j(Z_j) &= 0, \text{ if } x_{ij} > 0 \\ &\geq 0, \text{ if } x_{ij} = 0 \end{aligned} \tag{1}$$

for all $(i, j) \in I \times J$. The total supply Y_i at node i and the total demand Z_j at node j satisfy

$$Y_i = \sum_{j \in J} x_{ij}, \ \forall i \in I, \quad Z_j = \sum_{i \in I} x_{ij}, \ \forall j \in J, \tag{2}$$

which may be interpreted as conservation of flow constraints at the nodes of a bipartite network. By convention, we suppose that the amount of goods exchanged or

equivalently the internodal flow between any two nodes is nonnegative. Therefore,

$$x_{ij} \geq 0, \quad \forall\, (i,j) \in I \times J. \tag{3}$$

Samuelson [17] and Takayama and Judge [19], [20] show that this model has an equivalent minimization formulation. The equilibrium conditions (1) may be interpreted as the Karush–Kuhn–Tucker optimality conditions for the minimization problem:

$$\min \sum_{i \in I} \int_0^{Y_i} S(t)\, dt + \sum_{i \in I} \sum_{j \in J} \int_0^{x_{ij}} c_{ij}(t)\, dt - \sum_{j \in J} \int_0^{Z_j} I(t)\, dt \tag{4}$$

subject to the constraints (2) and (3). Because the cost functions are continuous and monotone, the objective function is a sum of convex functions. Therefore the objective function, which we denote by f, is itself a convex function and is also separable. Florian and Los [2] note that the quantities supplied Y_i, and the quantities consumed Z_i, are determined endogenously. Thus, (2) may be interpreted as a definition. Indeed, if the internodal flows are considered as variables, then (2) and (4) are equivalent to:

$$\min \sum_{i \in I} \int_0^{\sum_{j \in J} x_{ij}} S(t)\, dt + \sum_{i \in I} \sum_{j \in J} \int_0^{x_{ij}} c_{ij}(t)\, dt - \sum_{j \in J} \int_0^{\sum_{i \in I} x_{ij}} I(t)\, dt. \tag{5}$$

subject to the constraint (3). The latter is a nonlinear programming problem requiring the minimization of a convex, real-valued function $f(x)$ of the $nm \times 1$ vector x, subject to nonnegativity constraints on the nm variables. The standard technique for minimizing such a convex function is to employ a descent algorithm with an iterative search for the unique global minimum point x^* (see Appendix A for the general framework of a descent algorithm for unconstrained minimization). The sequence of iterates, which are successive approximations of x^*, will be denoted $x^0, x^1, \ldots$ or in general x^k.

In section 2 we discuss general Newton (second derivative) methods for the solution of nonlinear optimization problems and in section 3 we present the theoretical concepts which are important for the formulation of numerical algorithms.

2. Iterative Minimization Methods. We recall that a close relationship exists between the solution of a nonlinear system of equations and the minimization of a real-valued function. Given a vector-valued function $F(x)$ of n variables, let $F(x) = 0$ represent a general, possibly nonlinear system of n equations in n unknowns. In

principle, a solution x^* of this equation may be obtained by applying a fixed–point iteration. A classic example of a fixed–point iteration is the n–dimensional Newton method:

$$x^{k+1} = x^k - F'(x^k)^{-1}F(x^k), \quad k = 0, 1, \ldots, \tag{6}$$

where $F'(x^k)$ is the Jacobian of the function F evaluated at the point x^k. The well known theorem of Kantorovich [13] establishes the conditions under which iteration (6) converges quadratically and thus Newton's method is theoretically attractive. In a practical implementation of the iteration, the inverse of $F'(x^k)$ would not of course be explicitly computed. An appropriate alternative is to solve a linear system of equations. Indeed, suppose that we find the solution p^k of the linear system:

$$F'(x^k)p^k = -F(x^k). \tag{7}$$

It then follows that the Newton iteration (6) is equivalent to $x^{k+1} = x^k + p^k$. We note that the Newton method may be difficult to use in practice as the linear system (7) must be solved at each step.

Now consider the unconstrained minimization of a real–valued function $f(x)$ of n variables:

$$\min_x f(x). \tag{8}$$

Suppose that the first and second derivatives of f with respect to the vector x at x^k exist, and let $f_k(x)$ be the quadratic function obtained by approximating f by the first three terms of its Taylor series:

$$f_k(x) = f(x^k) + f'(x^k)(x - x^k) + \frac{1}{2}(x - x^k)^T f''(x^k)(x - x^k). \tag{9}$$

The derivatives are the gradient vector $g(x^k) = f'(x^k)^T$ and the symmetric Hessian matrix $G(x^k) = f''(x^k)$ respectively. A critical point $p^k = x - x^k$ of (9) that minimizes $f_k(x)$ must therefore satisfy the following equations.

$$G(x^k)p^k = -g(x^k) \tag{10}$$

We note that this is Newton's method for the equation $f'(x)^T = 0$, and therefore (10) is equivalent to (7) when we set $F(x) = g(x)$. If the Hessian $G(x^k)$ is positive definite, then the vector p^k is the unique global minimum of (9) and $x^{k+1} = x^k + p^k$ is taken as the next iterate. Linear system (10) is called the Newton equation and the vector p^k is said to be the Newton search direction toward the minimum point x^*.

A descent algorithm in which (10) is solved at each step can be extended to problems having simple bound constraints by employing an active set strategy. This technique requires the computation of a null basis matrix Z_k, corresponding to constraints exactly satisfied at the k–th iteration. In the case of simple bound constraints these 'active' constraints are represented by a matrix Z_k whose columns are taken from the $n \times n$ identity matrix I_n. The 'projected' Newton equations now contain the projected gradient $Z_k^T g(x^k)$ and the projected Hessian $Z_k^T G(x^k) Z_k$:

$$Z_k^T G(x^k) Z_k p^k = -Z_k g(x^k).$$

A direct numerical method for solving (7), such as Gaussian elimination, will require $O(n)$ storage for the vector $F(x^k)$ and $O(n^2)$ storage for the matrix $F'(x^k)$. If this matrix is large and sparse, then an LU factorization technique designed to limit fill–in may be appropriate. Thus, in many cases, direct numerical methods which preserve sparsity may be employed. In the event that the matrix $F'(x^k)$ is not explicitly available or is difficult to compute, then elements of the matrix may be approximated by a finite differencing scheme. An alternative to a direct approach for solving either (7) or (10) is to employ linear iterative methods. Such techniques are doubly iterative as they consist of a primary (outer) nonlinear iteration and a secondary (inner) linear iteration. Ortega and Rheinbolt [13] refer to this as a general approach for combining nonlinear methods with linear iterations in order to form composite methods. For example, successive overrelaxation (SOR) is an iterative scheme which is effectively employed in the solution of linear systems of equations arising in the numerical solution of partial differential equations. A composite method combining a primary nonlinear Newton iteration with a secondary linear SOR iteration to solve (7) is referred to as a composite *Newton–SOR* iteration. Sherman [18] analyzes the use of Newton–SOR techniques in the solution of nonlinear systems of equations and his results indicate that the number of inner linear iterations required to achieve better than linear convergence may increase exponentially with each outer nonlinear iteration.

For the minimization of nonlinear functions, the accurate determination of a Newton step p^k should not be crucial when the search is far from a minimum point as the Newton equations are derived from a Taylor series approximation near x^*. During the early stages of the iterative search an approximate search direction should be sufficient to reduce the value of the objective function f. An accurate computation of p^k becomes important as we approach the minimum x^*, where the Hessian matrix

$G(x^k)$ will converge to $G(x^*)$. Thus for large–scale problems a practical approach is to approximately solve the Newton equations by employing an iterative method with a variable tolerance on the residual error. The term 'truncated–Newton' is used to describe a minimization method in which a search direction is computed by the approximate solution of the Newton equations.

3. Truncated–Newton Methods. If the Hessian matrix $G(x^k)$ is large and possibly sparse, then conjugate gradient (CG) type methods employing matrix–vector products of the form Gv may be appropriate [1] for solving the Newton equations (10). When an explicit form of the Hessian matrix is available this product can be computed directly. Otherwise, if it is expected that $G(x^k)$ will be sparse, a finite differencing scheme can be applied to approximate the Hessian. A third approach, described in [3], is a finite difference approximation related to the Gateaux derivative of the gradient vector

$$G(x^k)v \approx \frac{g(x^k + hv) - g(x^k)}{h}.$$

The approximate solution of the Newton equations by employing a restricted number of 'inner' linear conjugate gradient iterations is proposed by Dembo and Steihaug in [1]. Upon termination of the inner iteration the final iterate is then used as the search direction p^k. In theory, a single conjugate gradient iteration produces the steepest descent direction $p^k = -g(x^k)$, whereas n iterations produce the Newton direction. When the number of iterations is less than n an approximation to the Newton direction is produced. The classical Newton method takes a step $x^{k+1} = x^k + p^k$ towards the minimum. However, the Taylor series approximation of f will be inaccurate far from a minimum point x^*. Another approach that is often applied in practice is to perform a parameterized line search to insure a decrease in f. In this case a positive constant $0 < \alpha \leq 1$ is found, such that $f(x^{k+1}) < f(x^k + \alpha p^k)$.

Given the residual $r_j = G(x^k)p^j + g(x^k)$, at the j–th inner iteration of the linear algorithm, it is shown in [1] that $\|r_j\|_2 / \|g(x^k)\|_2 \to 0$ is a necessary and sufficient condition for a truncated–Newton algorithm to converge superlinearly. Therefore, they suggest that a suitable termination criterion at the k–th nonlinear outer iteration is

$$\frac{\|r_j\|_2}{\|g(x^k)\|_2} \leq \min \{ 1/k, \|g(x^k)\|_2 \}.$$

Unfortunately, it is known that the numerical values of the residual norm $\|r_j\|_2$ may not decrease monotonically due to a loss of orthogonality caused by rounding errors in

conjugate gradient type algorithms. In fact, the residual may increase over the course of several iterations before continuing to decrease once again [15]. Therefore, a large residual norm does not necessarily indicate that the search direction is inaccurate. An alternative termination criteria can be derived by first defining the quadratic function $Q(p)$:

$$Q(p) = p^T g(x) + \frac{1}{2} p^T G(x) p,$$

which is a second order approximation of $f(x+p) - f(x)$. This function is minimized at each nonlinear iteration by the conjugate gradient algorithm and its value decreases monotonically during the inner linear iterations. An alternative test [12] for accepting a search direction is given by

$$j \, (\, Q_j - Q_{j-1} \,) \, / \, Q_j \leq \beta,$$

where $Q_j = Q(p^j)$ at the j–th inner iteration. The parameter β is chosen to insure a particular convergence rate. For example, $\beta = 0.5$ results in at least a linear rate of convergence for the nonlinear method, with rate constant 0.5. In practice, as the minimum point is approached, such a Newton type method begins to converge quadratically.

We now consider several numerical details of conjugate gradient type linear iterative solvers. Let A be a symmetric positive definite matrix of dimension $n \times n$ and let b be a known $n \times 1$ vector. The conjugate gradient method, originally due to Hestenes and Stiefel [5], can be viewed as a direct method for solving the linear system $Ax = b$, which theoretically terminates after n steps for any positive definite matrix. In practice, however, full precision may not be achieved in n steps because of rounding errors and thus Reid [15] suggests that the algorithm be applied iteratively. When A is indefinite, the conjugate gradient algorithm may fail due to a division by zero. Several algorithmic variations to overcome this difficulty are possible and they rest upon the mathematical equivalence of the Lanczos iteration with the conjugate gradient algorithm for solving linear systems of equations. The Lanczos method [4] for solving the symmetric eigenvalue problem is equivalent to a Galerkin projection onto the Krylov subspace:

$$K_k = \{ \, v_1, \, Av_1, \, \ldots, \, A^{k-1} v_1 \, \}.$$

The conjugate gradient algorithm is a Galerkin method for solving the linear system $Ax = b$, over the Krylov subspace K_k with $v_1 = r_0 / \|r_0\|_2$, where $r_0 = b - Ax_0$ and

x_0 is the initial linear iterate [16]. In the Lanczos method one computes T_k, the symmetric tridiagonal matrix representation of the orthogonal projection of A onto K_k. Moreover, the linear conjugate gradient algorithm corresponds to the indirect application of Gaussian elimination *without* pivoting to solve a linear system having coefficient matrix T_k. It is for this reason that the algorithm is potentially unstable, as a zero pivot may be encountered when A, and therefore T_k, are indefinite. One effective way of avoiding this problem is given by the SYMMLQ algorithm [14], which employs a stable LQ factorization of the tridiagonal matrix T_k. However, in nonlinear optimization problems, the solution of an indefinite system may not correspond to a descent direction. An alternative approach described in Gill *et al* [3] is to compute the stable factorization of a nearby positive definite matrix. Nash [10] applies such a 'modified' factorization scheme to the matrix T_k produced by the Lanczos method in order to derive a numerically stable inner linear iteration.

If we ignore the finite termination requirement, the conjugate gradient algorithm has a linear convergence rate proportional to $(\kappa^{1/2} - 1)/(\kappa^{1/2} + 1)$, where κ is the L_2 condition number of the matrix A [7]. A standard technique employed to accelerate the iterative solution of the linear system $Ax = b$ is to apply a preconditioning transformation. Indeed, suppose that x is obtained by solving the system $B^{-T}AB^{-1}y = B^{-T}b$, where we define the matrix $M = B^TB$. The matrix $B^{-T}AB^{-1}$ has the same eigenvalue spectrum as $M^{-1}A$, and M can be designed so that $\kappa_2(M^{-1}A) < \kappa_2(A)$ thus improving the linear convergence rate. Given an initial vector v_1 such that $\|v_1\|_M = 1$, the preconditioned Lanczos [4] method consists of the three–term recurrence relation:

$$\beta_{j+1}Mv_{j+1} = (\, A - \alpha_j M\,)v_j - \beta_j Mv_{j-1}, \quad \alpha_j = v_j^T Av_j,$$

where $v_0 \equiv 0$. Initially, we choose $\beta_1 = \|M^{-1}b\|_M$, and $v_1 = M^{-1}b/\beta_1$. The scalar $\beta_{j+1} \geq 0$ is chosen so that $\|v_{j+1}\|_M = 1$. (We recall that the norm $\|\cdot\|_M$ is defined by $\|v\|_M^2 = v^TMv$.) After the k–th linear iteration we have

$$AV_k = MV_kT_k + \beta_{k+1}Mv_{k+1}e_k^T,$$

where T_k is the symmetric tridiagonal matrix described in the previous paragraph, $V_k^TAV_k = T_k$ and $V_k^TMV_k = I_k$. If we substitute x_k, the k–th iterate, into the original linear system, we see that $Ax_k = b$ is transformed into $V_k^TAV_ky_k = V_k^Tb$. Therefore, it follows that $T_ky_k = \beta_1e_1$ and $x_k = V_ky_k$.

In the application of the Lanczos method to $Ax = b$, we might wish to update an approximation x_k of the solution x at each (inner) iteration. This can be achieved by

employing a recursive form of the Cholesky factorization for the tridiagonal matrix T_k. If the matrix A is positive definite, then so is $T_k = V_k^T A V_k$. Therefore, the matrix T_k has the factorization $T_k = L_k D_k L_k^T$, where D_k is diagonal with positive elements d_j. The matrix L_k is lower bidiagonal with unit diagonal and subdiagonal elements μ_j. Given $d_1 = \alpha_1$, we have the following recurrence relations:

$$\mu_{j-1} = \beta_j / d_{j-1}, \quad d_j = \alpha_j - \beta_j \mu_{j-1}, \quad j = 2, \ldots, k.$$

Since the vector y_k changes completely as k increases, it is not practical to compute $V_k y_k$ at each iteration. However, x_k can still be computed iteratively. Given the vector $p_k = L_k^T y_k$ and the matrix $C_k = V_k L_k^{-T}$, it follows that

$$L_k D_k p_k = \beta_1 e_1, \quad x_k = C_k p_k.$$

By the relation $C_k L_k^T = V_k$ we then have

$$C_k = [\, C_{k-1}, \; c_k \,], \quad c_k = v_k - \mu_{k-1} c_{k-1}.$$

If we define $p_k = [\, \rho_1, \; \rho_2, \; \ldots, \; \rho_k \,]$, the relation $L_k D_k p_k = \beta_1 e_1$ gives

$$p_k^T = [\, p_{k-1}^T, \; \rho_k \,]^T, \quad \rho_k = -\mu_{k-1} d_{k-1} \rho_{k-1} / d_k = -\beta_{k-1} \rho_{k-1} / d_k.$$

Finally, the next linear iterate is computed from the recurrence relation:

$$x_k = C_k p_k = C_{k-1} p_{k-1} + \rho_k c_k = x_{k-1} + \rho_k c_k.$$

If the initial linear iterate is $x_0 = 0$, then the iteration given above is theoretically equivalent to the linear (preconditioned) conjugate gradient algorithm for solving $Ax = b$ (see Appendix A for a description of both algorithms or chapter 9 of Golub and Van Loan [4]). An estimate of the residual error may be derived as follows. First,

$$\begin{aligned} A x_k &= A V_k y_k = M V_k T_k y_k + \beta_{k+1} M v_{k+1} e_k^T y_k \\ &= \beta_1 M V_k e_1 + \beta_{k+1} M v_{k+1} e_k^T y_k. \end{aligned}$$

Therefore, since $\beta_1 M v_1 = b$, we have

$$\| A x_k - b \|_2 = \beta_{k+1} \| M v_{k+1} \|_2 \, | \, e_k^T y_k \, |.$$

When the above linear iteration is carried out within each outer nonlinear iteration, it is equivalent to the implicit M–orthogonalization of the set K_k of Krylov vectors by the Gram–Schmidt process.

For certain problems, we have observed a severe loss of numerical precision when solving $Ax = b$ by the preconditioned Lanczos iteration. Numerical experiments with the Lanczos algorithm indicate that the residual error in solving the Newton equations is much larger when A and consequently M are ill–conditioned. Recall that we are effectively solving a tridiagonal system:

$$x_k = V_k T_k^{-1}(\beta_1 e_1).$$

If the eigenvalues of T_k differ greatly in magnitude, then there can be a loss of significant figures in the computed solution. A measurable effect is the growth of the computed vector $\bar{f}_j \equiv M\bar{v}_j$ due to numerical cancellation. The resulting loss of significant figures in the residual is illustrated by the following example.

We solved a sequence of linear systems $Ax = b$ of progressively larger dimension. The matrix A contains elements of a well known and extremely ill–conditioned Hilbert matrix H_n, $a_{ij} = 1/(i + j - 1)$ for $|i - j| \leq 2$ and $a_{ij} = 0$ for $|i - j| > 2$. The vector b was chosen as $b = Ae$, where $e = (1, \ldots, 1)^T$. An Incomplete Cholesky (ICCG) preconditioning strategy [4] for computing B was applied. Table 1 indicates the minimum attainable residual error for each system. Computations were performed in double precision floating point arithmetic on a computer with unit rounding error $u = 2^{-52} \approx 2.2 \times 10^{-16}$. These effects are, however, not observed when the preconditioned

n	$\kappa_2(A)$	$\|Ax_k - b\|_2$
10	1.2×10^5	3.9×10^{-14}
20	3.2×10^8	3.1×10^{-12}
30	1.4×10^{11}	1.7×10^{-9}
40	1.7×10^{12}	4.1×10^{-4}
50	1.3×10^{17}	3.9×10^{-1}

Table 1: Residual Error for Preconditioned Lanczos

conjugate gradient algorithm is employed.

For optimization problems of large dimension it may not be practical to explicitly form a preconditioning matrix, which in theory serves as an approximation to the inverse of the Hessian matrix. Recall that the preconditioning step of the Lanczos iteration corresponds to the solution of $M\bar{u}_j = w_j$ for the vector $\bar{u}_j$ (see Appendix A). Nash [11] proposes that an effective preconditioner can be constructed by employing

limited–memory quasi–Newton optimization techniques [6]. The matrix M^{-1} can be represented by H_k, an approximation to the inverse Hessian, where we have

$$p^k = -H_k g^k.$$

In particular, local curvature information at the k–th nonlinear iteration is represented by the difference vectors $\{ y_k, s_k \}$,

$$y_k = g^{k+1} - g^k, \;\; s_k = x^{k+1} - x^k.$$

The BFGS formula [3] for the inverse Hessian is then given by:

$$H_{k+1} = V_k^T H_k V_k + \rho_k s_k s_k^T, \;\; k = 0, 1, 2, \ldots$$

where $\rho_k = 1/y_k^T s_k$, and

$$V_k = I - \rho_k y_k s_k^T.$$

A two–stage preconditioning strategy can be based on a diagonal scaling of the Hessian matrix followed by several low rank BFGS updates. Curvature information is obtained by employing a restricted number of the vectors $\{ y_k, s_k \}$ stored during previous iterations of the nonlinear optimization algorithm.

4. A Newton Block–SOR Method. Ortega and Rheinboldt [13] describe composite iterative methods in which a linear SOR iteration is extended to serve as the primary 'outer' nonlinear iteration and Newton's method is employed as the secondary 'inner' iteration. Given a nonlinear system of equations $F(x) = 0$ of s variables, with components $F_1, \ldots, F_s$, the i–th equation

$$F_i(x_1^k, \ldots, x_{i-1}^k, x_i, x_{i+1}^{k-1}, \ldots, x_s^{k-1}) = 0 \tag{11}$$

can be solved for x_i and then we set $x_i^k = x_i$. It follows that a one–step *SOR–Newton* iteration takes the form:

$$x_i^{k+1} = x_i^k - \omega F_i(x^{k,i}) / \partial_i F_i(x^{k,i}), \;\; i = 1, \ldots, s, \;\; k = 0, 1, \ldots, \tag{12}$$

where ω is a relaxation parameter and where we have adopted the notation:

$$x^{k,i} = (x_1^k, \ldots, x_{i-1}^k, x_i, x_{i+1}^{k-1}, \ldots, x_s^{k-1})^T. \tag{13}$$

Nagurney [9] and Zubieta [21] propose that a block generalization of an SOR–Newton method can be applied to both linear and nonlinear spatial price equilibrium problems. Zubieta [21] incorporates an active set strategy for solving a nonlinear and separable SPEP.

Marcotte *et al* [8] solve the spatial price equilibrium problem by an approach which takes advantage of the inherent bipartite graph structure. Their method is based on the linearization of the cost functions in (5) at each 'outer' nonlinear iteration of the minimization algorithm. These linearized subproblems are solved by a very efficient block–SOR scheme. The authors cite two primary motivations for their approach. First, previous computational experience has shown that an SPEP with linear cost functions may be solved analytically with a block–SOR approach. Second, they propose that sparse, separable problems are best solved by a combination of SOR and active set strategies. This last observation is also based on the property that, when the transportation costs are linear, the solution will have at most $m+n-1$ positive transshipment (x_{ij}) variables. Here, their approach results in a much smaller problem to be solved at each nonlinear iteration.

Given the objective function $f(x)$ of nm variables $x = (x_{ij})_{i\in I, j\in J}$, the algorithm proposed by Marcotte *et al* [8] can be summarized as follows. At the k–th outer nonlinear iteration one solves a linearized problem defined by an objective function $\bar{f}(x)$ having supply, demand and transportation costs given by:

$$\begin{aligned} \bar{S}_i(Y_i) &= S_i(Y_i^k) + S'(Y_i^k)(Y_i - Y_i^k) \\ \bar{c}_{ij}(x_{ij}) &= c_{ij}(x_{ij}^k) + c'_{ij}(x_{ij}^k)(x_{ij} - x_{ij}^k) \\ \bar{I}_j(Z_j) &= I_j(Z_j^k) + I'(Z_j^k)(Z_j - Z_j^k). \end{aligned} \tag{14}$$

In other words, one finds x_{ij}, Y_i and Z_j such that

$$\begin{aligned} \bar{S}_i(Y_i) + \bar{c}_{ij}(x_{ij}) - \bar{I}_j(Z_j) &= 0, \text{ if } x_{ij} > 0 \\ &\geq 0, \text{ if } x_{ij} = 0 \end{aligned} \tag{15}$$

subject to the constraints (2) and (3). A line search can then be performed to insure a decrease in the objective function f at the next iterate x^{k+1}.

An iterative block–SOR (*BSOR*) scheme described in [8] is employed to solve each linearized subproblem. This technique can be viewed as an extension of composite k–step Newton–SOR techniques for solving nonlinear systems of equations (see note 7.4–10 in Ortega and Rheinboldt). If the i–th block of variables is denoted $X_i = (x_{ij})_{j\in J}$,

the l–th iteration of the *BSOR* algorithm corresponds to the minimization problem:

$$\min_{X_i} \tilde{f}(X_1^l, \ldots, X_{i-1}^l, X_i, X_{i+1}^{l-1}, \ldots, X_m^{l-1}) \tag{16}$$

for the block of variables X_i, with all other variables fixed, and then setting $X_i = X_i^l$. It is observed by Zubieta [21] that (16) can be solved in closed form provided that the active constraints (nonzero variables) are known *a priori*. In practice, the standard technique is to employ an active set strategy. However, this requires that the strict complementarity condition

$$x_{ij}^* [\, S_i(Y_i^*) + c_{ij}(x_{ij}^*) - I_j(Z_j^*) \,] = 0$$

holds at a solution (x^*, Y^*, Z^*). Marcotte *et al* [8] found in numerical tests that degeneracy can occur in the traditional active set implementation. Nevertheless, the authors demonstrate the effectiveness of an alternative active set algorithm, referred to as *AS*, which they prove to converge in a finite number of steps.

A linearized subproblem (14) should only represent a good approximation to the initial nonlinear subproblem (1) when the iterative search is close to an equilibrium solution x^*. Therefore the accurate solution of (16) should not be crucial far from this minimum point. It follows that the number l of inner iterations can be increased as the outer nonlinear iteration approaches the minimum point. The strategy proposed by the authors of [8] is to increase the number of inner block–SOR iterations at the k–th outer nonlinear iteration by the simple formula $l_k = 1 + k$. If L iterations of the block–SOR algorithm are performed, it is possible to show [8] that such a composite L–step Newton block–SOR (*NBSOR*) scheme converges at least linearly to a stationary point (see Ortega and Rheinboldt, p. 321).

5. Numerical Results. We summarize numerical experiments performed on randomly generated test problems proposed by Marcotte *et al* [8]. The problems vary in size and all consist of supply and demand functions of the form:

$$\begin{aligned} S_i(Y_i) &= \xi_i + (Y_i/\eta_i)^2 \\ I_j(Z_j) &= \frac{1}{\theta_j} \ln\left(\frac{2000}{Z_j}\right), \end{aligned}$$

where ξ_i, η_i and θ_j are pseudo–random variables on the intervals:

$$\xi_i \in [\,3,\, 5\,], \quad \eta_i \in [\,15,\, 20\,], \quad \theta_j \in [\,0.1,\, 0.5\,].$$

We compare the running times of an implementation of the L–step *NBSOR* algorithm with a truncated–Newton code *TN* written by Nash [11] and modified in such a way that the preconditioned Lanczos iteration described in section 3 is used (as opposed to the standard preconditioned conjugate gradient formulation). A consistent basis for comparison was maintained by employing single precision implementations of both algorithms on the SUN 3/50 and SUN Sparc/1 computers. The unit rounding error for both machines in single precision floating point arithmetic is $u = 2^{-23}$.

The convergence test for the 'outer' nonlinear iteration of the *NBSOR* algorithm relies strictly on the L_1 vector norm of the gradient $g(x) = f'(x)^T$,

$$\|g(x)\|_1 = \sum_{i \in I} \sum_{j \in J} | S_i(Y_i^k) + c_{ij}(x_{ij}) - I_j(Z_j) |,$$

with an error tolerance of $\|g(x)\|_1 \leq 10^{-5}$. The stopping criteria for *TN* are described in Chapter 8 of Gill *et al* [3]. The CPU times reported for the *TN* code reflect the amount of computation required to obtain either 5 significant figures in the objective function or the gradient at the minimum point.

Experiments with the *TN* code were also run in double precision on both machines, where $u = 2^{-52} \approx 2.2 \times 10^{-16}$. These computations are considered as 'exact' results for comparison with the single precision runs. In Figure 1 we have plotted $\log_{10}(E)$ of the relative error E in the objective function versus CPU time for a problem with $n = 20$, or in other words 400 variables. This graph illustrates the CPU time required to obtain a fixed number of significant figures in the objective function. We note that the convergence is fairly rapid up to the last significant figure. Gill *et al* [3] (see pp. 314–15) note that this behaviour is to be expected when approaching the limiting accuracy of the computation, in particular for Newton–type methods.

Table 2 contains a summary of the results for small size problems (run on the SUN 3/50) containing linear transportation cost functions of the form

$$c_{ij}(x_{ij}) = \gamma_{ij} x_{ij},$$

where the coefficients γ_{ij} are uniformly distributed on the interval $\gamma_{ij} \in [\, 5,\ 10\,]$. Results for a second set of medium size problems with linear transportation costs (run on the SUN 3/50) are reported in Table 3. The computation times for both the *NBSOR* and *TN* codes are comparable. The third set of problems (run on the SUN Sparc/1) are reported in Table 4. These contain convex nonlinear transportation cost functions

$$c_{ij}(x_{ij}) = (x_{ij}/\gamma_{ij})^4$$

where the γ_{ij} are again uniformly distributed on [5, 10]. Results for problems of mixed sizes are provided.

In all cases for the random test problems, condition estimates for the Hessian and preconditioner indicated that these matrices were all well–conditioned. Thus we did not observe any loss of significant figures in the computed linear residuals.

6. Conclusion. We have described a general second derivative based optimization algorithm for solving the spatial price equilibrium problem. On the basis of numerical tests we conclude that a truncated–Newton method can compete successfully with more specialized methods which exploit the structure of the optimal solutions in a bipartite graph when the transportation costs are linear as well as the inherent bipartite graph structure of the problem. This is particularly evident for the nonlinear transportation cost problems, where the active set strategy is not likely to be advantageous. Although the truncated–Newton method requires computing times which are just comparable to a Newton Block–SOR scheme to achieve a given precision for problems having linear transportation costs, appreciable savings can be attained for problems with nonlinear transportation costs. For nonlinear costs it also appears that the Newton–SOR scheme is limited by a linear convergence rate. We note that for a Newton type minimization method, when the Hessian matrix is too large or its sparsity structure can not be exploited, an effective alternative to directly solving the Newton equations is to employ an iterative linear solver. Various approaches related to the linear conjugate gradient algorithm rely strictly on matrix–vector products. These products may be approximated by a finite differencing scheme, and thus the explicit formation of the Hessian is not required. A judicious choice of linear iterative method for approximately solving the Newton equations avoids the potential numerical difficulties associated with an indefinite or ill–conditioned Hessian matrix.

Acknowledgements. Professor Lourdes Zubieta of Bishop's University kindly provided us with the source code for her *NBSOR* program. Dr. Omar Drissi–Kaïtouni of CRT was able to pass along many insights from his experience with network equilibrium problems. Finally, we would like to thank Professor Stephen G. Nash of George Mason University for providing us with an early version of his truncated–Newton minimization package.

Appendix A. Algorithms.

Descent Algorithm for Unconstrained Minimization.

Given initial estimate $x^{(0)}$.
for $k = 0, \ldots$ **do**
(1) If $x^{(k)}$ satifies convergence criteria, then stop.
(2) Compute $p^{(k)}$ such that ${p^{(k)}}^T g^{(k)} < 0$.
(3) Find an $\alpha^{(k)} > 0$ so that $f(x^{(k)} + \alpha^{(k)} p^{(k)}) < f(x^{(k)})$.
(4) Set $x^{(k+1)} = x^{(k)} + \alpha^{(k)} p^{(k)}$
end

Preconditioned Conjugate Gradients.

$x_0 = 0$, $r_0 = b$, $p_0 = M^{-1} r_0$, $z_0 = p_0$
for $k = 0, \ldots$ **do**
(1) **solve** $M z_{k+1} = r_{k+1}$
(2) $\alpha_k = r_k^T z_k / p_k^T A p_k$
(3) $x_{k+1} = x_k + \alpha_k p_k$
(4) $r_{k+1} = r_k - \alpha_k A p_k$
(5) $\beta_k = r_{k+1}^T z_{k+1} / r_k^T z_k$
(6) $p_{k+1} = z_{k+1} + \beta_k p_k$
end

Preconditioned Lanczos.

$\beta_1 = \|M^{-1}b\|_M$, $v_1 = M^{-1}b/\beta_1$, $u_1 := Av_1$, $f_1 = Mv_1$
for $j = 1, 2, \ldots$ **do**
  $\alpha_j := v_j^T u_j$
  $w_j := u_j - \alpha_j f_j$
  solve $M\bar{u}_j = w_j$
  if $j = 1$ **then**
    $d_1 := \alpha_1$, $c_1 := v_1$, $\rho_1 = \beta_1/\alpha_1$, $x_1 := \rho_1 v_1$
  else
    $\mu_{j-1} := \beta_j/d_{j-1}$, $d_j := \alpha_j - \beta_j\mu_{j-1}$
    $c_j := v_j - \mu_{j-1}c_{j-1}$, $\rho_j := -\beta_{j-1}\rho_{j-1}/d_j$
    $x_j := x_{j-1} + \rho_j c_j$
  end
  $\beta_{j+1} := (\bar{u}_j^T w_j)^{1/2}$
  if $\beta_{j+1} \leq tol$ **then** *stop*
  $v_{j+1} := \bar{u}_j/\beta_{j+1}$
  $f_{j+1} := w_j/\beta_{j+1}$
  $u_{j+1} := Av_{j+1} - \beta_{j+1}f_j$
end

References

[1] R. S. DEMBO and T. STEIHAUG. "Truncated-Newton algorithms for large-scale unconstrained optimization", *Math. Prog.*, **26** (1983), pp. 190–212.

[2] M. FLORIAN and M. LOS. "A new look at static spatial price equilibrium models", *Reg. Sci. Urban Econ.*, **12** (1982), pp. 175–82.

[3] P. E. GILL, W. MURRAY and M. H. WRIGHT. *Practical Optimization*, Academic Press, London, 1981.

[4] G. H. GOLUB and C. VAN LOAN. *Matrix Computations*, 2nd Ed., The Johns Hopkins University Press, Baltimore, Maryland, 1990.

[5] M. HESTENES and E. STIEFEL. "Methods of conjugate gradients for solving linear systems", *J. Res. Nat. Bur. Standards*, **49** (1952), pp. 409–36.

[6] D. C. LIU and J. NOCÉDAL. "On the limited memory BFGS method for large scale optimization", *Math. Prog.*, **45** (1989), pp. 503–28.

[7] D. G. LUENBERGER. *Linear and Nonlinear Programming*, Addison-Wesley, New York, 1984.

[8] P. MARCOTTE, G. MARQUIS and L. ZUBIETA. "A Newton-SOR method for solving the spatial price equilibrium problem", Tech. Report CRT-679, Center for Research on Transportation, Université de Montréal, 1990.

[9] A. NAGURNEY. "Computational comparisons of Spatial Price Equilibrium methods", *J. Regional Science*, **27** (1987), pp. 57–76.

[10] S. G. NASH. "Newton-type minimization via the Lanczos method", *SIAM J. Numer. Anal.*, **4** (1984), pp. 770–88.

[11] S. G. NASH. "Preconditioning of truncated-Newton methods", *SIAM J. Sci. Stat. Comput.*, **3** (1985), pp. 559–616.

[12] S. G. NASH and A. SOFER. "Assessing a search direction within a truncated-Newton method", *Oper. Res. Letters*, **9** (1990), pp. 219–221.

[13] J. M. ORTEGA and W. C. RHEINBOLDT. *Iterative Solution of Nonlinear Equations in Several Variables*, Academic Press, New York, 1970.

[14] C. C. PAIGE and M. A. SAUNDERS. "Solution of sparse indefinite systems of linear equations" *SIAM J. Numer. Anal.*, **12** (1975), pp. 617–24.

[15] J. K. REID. "On the method of conjugate gradients for the solution of large sparse systems of linear equations", *Large Sparse Sets of Linear Equations*, Academic Press, New York, 1971, pp. 231–54.

[16] Y. SAAD. "Méthodes numériques pour la résolution de problèmes matriciels de grandes dimensions", Thèse de docteur d'état es–sciences, IMAG Grenoble, France, 1983.

[17] P. A. SAMUELSON. "Spatial price equilibrium and linear programming", *American Economic Review*, **42** (1952), pp. 283–304.

[18] A. H. SHERMAN. "On Newton–iterative methods for the solution of nonlinear equations", *SIAM J. Numer. Anal.*, **15** (1978), pp. 755–71.

[19] T. TAKAYAMA and G. G. JUDGE. "Equilibrium among spatially separated markets: a reformulation", *Econometrica*, **32** (1964), pp. 510–24.

[20] T. TAKAYAMA and G. G. JUDGE. "Alternative spatial equilibrium models", *J. Regional Science*, **10** (1970), pp. 1–12.

[21] L. ZUBIETA. "Modèles d'équilibre spatial et méthodes de résolution", Thèse, Département d'informatique et de recherche opérationnelle, Université de Montréal, 1986.

PROB	*NBSOR*		*TN*		
	outer	CPU (sec)	*outer*	*inner*	CPU (sec)
5.1	11	1.5	3	6	0.9
5.2	5	0.9	3	6	1.3
5.3	11	1.3	3	6	0.9
5.4	14	1.9	3	6	0.9
5.5	8	0.8	3	6	0.9
10.1	10	3.1	3	6	3.1
10.2	8	2.2	5	11	5.7
10.3	13	4.2	4	8	3.5
10.4	9	2.5	4	8	3.5
10.5	11	3.7	5	11	5.6
15.1	21	15.8	6	12	13.0
15.2	11	7.5	5	10	10.9
15.3	14	9.2	6	13	14.6
15.4	45	52.1	6	12	13.0
15.5	10	6.6	6	13	14.6
20.1	21	23.1	8	17	33.8
20.2	26	30.9	7	14	27.5
20.3	17	16.4	7	14	27.2
20.4	21	21.8	8	17	34.2
20.5	18	17.8	7	14	27.9

Table 2: Small size problems (SUN 3/50)

PROB	*NBSOR*		*TN*		
	outer	CPU (sec)	*outer*	*inner*	CPU (sec)
50.1	54	286.6	11	24	285.1
50.2	59	280.9	10	21	281.3
50.3	49	242.1	10	22	238.2
50.4	26	78.8	11	23	297.4
50.5	34	116.6	10	21	241.1

Table 3: Medium size problems (SUN 3/50)

PROB	*NBSOR*		*TN*		
	outer	CPU (sec)	*outer*	*inner*	CPU (sec)
10.1	14	0.3	12	51	0.6
10.2	20	0.5	24	66	0.9
10.3	53	3.6	36	92	1.2
10.4	44	2.5	20	80	1.0
10.5	53	3.4	31	70	1.0
15.1	112	29.8	20	62	1.6
15.2	116	29.9	32	80	2.2
15.3	120	31.6	36	77	2.3
15.4	53	5.8	12	36	1.0
15.5	139	45.3	66	255	6.3
20.1	169	107.4	61	235	10.4
20.2	38	4.9	19	61	2.9
20.3	160	99.7	48	141	6.6
20.4	91	28.5	22	84	3.7
20.5	262	216.3	100	358	16.0
30.1	646	> 60min	182	720	54.0
30.2	88	55.0	44	139	15.0

Table 4: Nonlinear Transportation Costs (SUN Sparc/1)

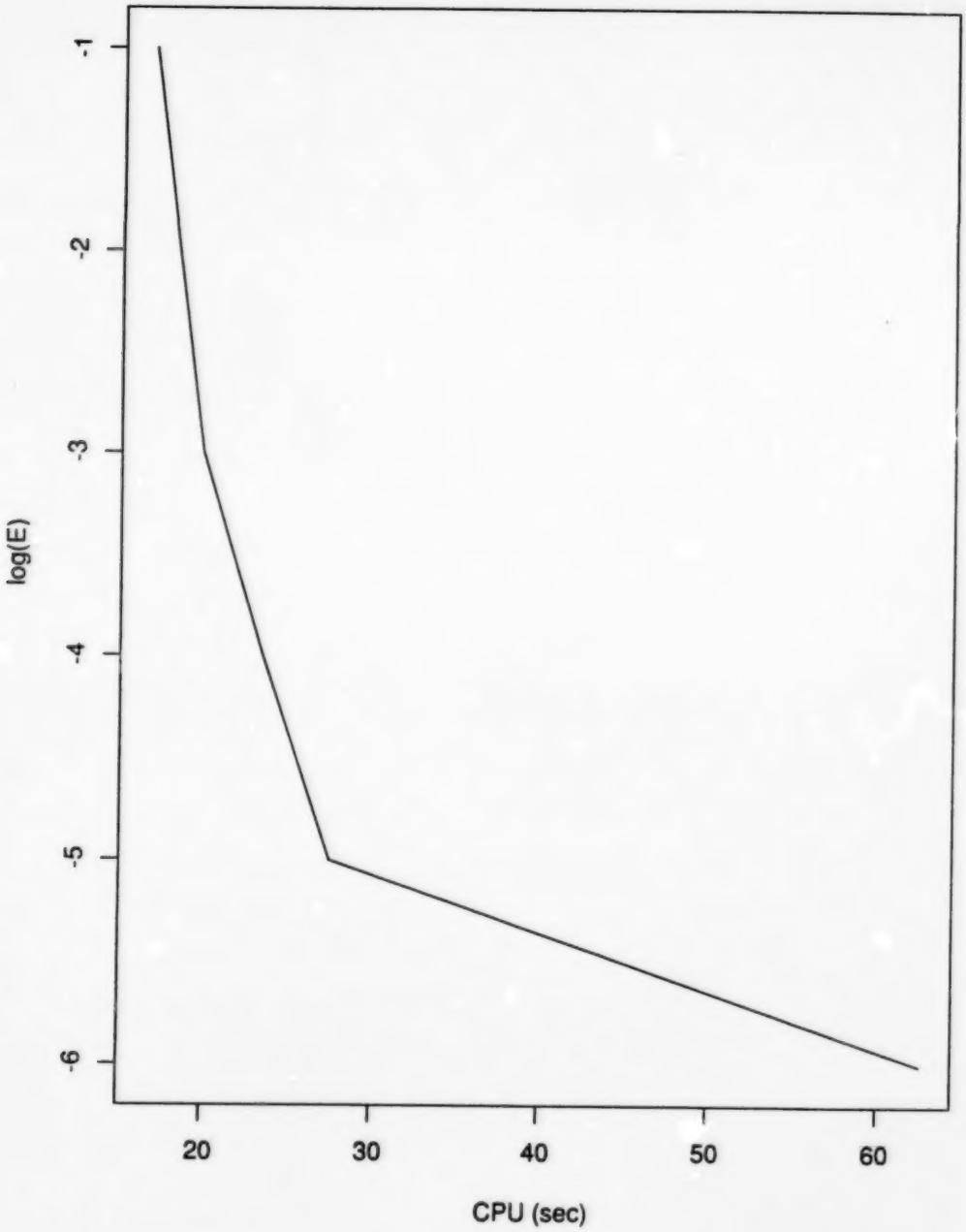

Figure 1: Precision vs CPU Time